N94-21476

A MULTIGRID METHOD FOR STEADY EULER EQUATIONS ON UNSTRUCTURED ADAPTIVE GRIDS

5/2-64
197572
p. 15

Kris Riemslagh and Erik Dick
Universiteit Gent
Sint-Pietersnieuwstraat 41, 9000 Gent
Belgium

SUMMARY

A flux-difference splitting type algorithm is formulated for the steady Euler equations on unstructured grids. The polynomial flux-difference splitting technique is used. A vertex-centered finite volume method is employed on a triangular mesh.

The multigrid method is in defect-correction form. A relaxation procedure with a first order accurate inner iteration and a second-order correction performed only on the finest grid, is used. A multi-stage Jacobi relaxation method is employed as a smoother. Since the grid is unstructured a Jacobi type is chosen. The multi-staging is necessary to provide sufficient smoothing properties.

The domain is discretized using a Delaunay triangular mesh generator. Three grids with more or less uniform distribution of nodes but with different resolution are generated by successive refinement of the coarsest grid. Nodes of coarser grids appear in the finer grids. The multigrid method is started on these grids. As soon as the residual drops below a threshold value, an adaptive refinement is started. The solution on the adaptively refined grid is accelerated by a multigrid procedure. The coarser multigrid grids are generated by successive coarsening through point removement. The adaption cycle is repeated a few times.

Results are given for the transonic flow over a NACA-0012 airfoil.

THE SPACE DISCRETIZATION

Flux-Difference Splitting

Figure 1 shows part of an unstructured triangular grid. In the vertex-centered finite volume method nodes are located at the vertices of the grid. Every node has a control volume, constructed by connecting the centers of gravity of the cells surrounding the node. To close the control volumes on the boundary, the midpoints of the boundary edges are chosen as vertices of the control volumes.

To define the flux through a side of a control volume, use is made of the flux-difference splitting

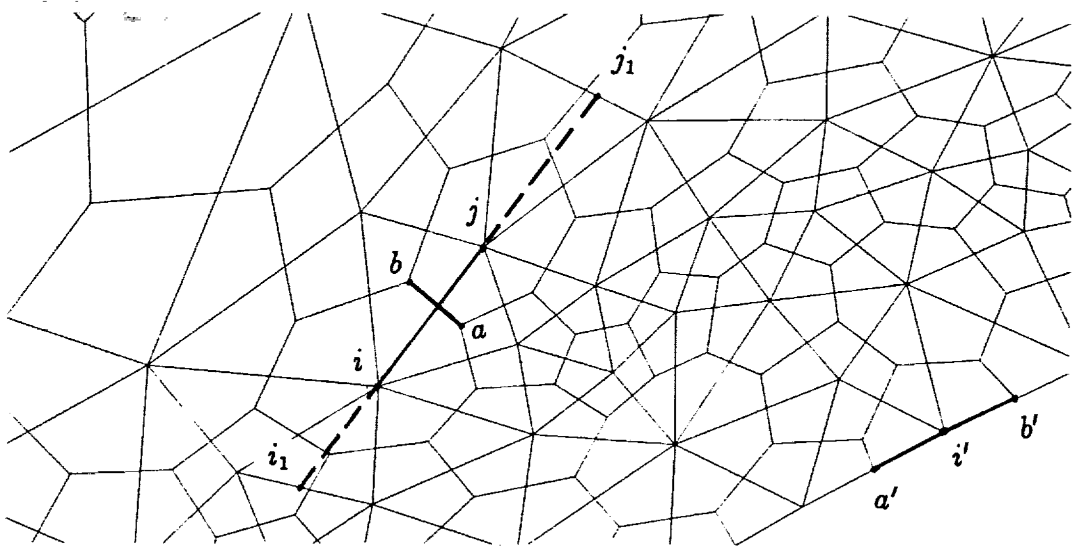

Figure 1: Vertex-centered discretization.

principle. The Euler equations in two dimensions take the form

$$\frac{\partial U}{\partial t} + \frac{\partial f}{\partial x} + \frac{\partial g}{\partial y} = 0, \tag{1}$$

where U is the vector of conserved variables, f and g the Cartesian flux vectors, given by

$$U = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{pmatrix}, f = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uH \end{pmatrix}, g = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vH \end{pmatrix}, \tag{2}$$

where ρ is the density; u and v are the Cartesian velocity components in the x and y directions, respectively; p is the pressure; $E = p/(\gamma - 1)\rho + u^2/2 + v^2/2$ is the total energy; $H = \gamma p/(\gamma - 1)\rho + u^2/2 + v^2/2$ is the total enthalpy; and γ is the adiabatic constant.

For the side ab of the node i and node j control volumes (figure 1), the flux-difference can be written as

$$\Delta F_{i,j} = (n_x \Delta f_{i,j} + n_y \Delta g_{i,j}) \Delta s_{i,j}, \tag{3}$$

where $\Delta f_{i,j}$ and $\Delta g_{i,j}$ denote the differences of the Cartesian flux vectors, n_x and n_y are the components of the unit normal to the side ab in the sense i to j, and $\Delta s_{i,j}$ is the length of the side. The differences of the Cartesian flux vectors are

$$\Delta f_{i,j} = f_j - f_i \ , \quad \Delta g_{i,j} = g_j - g_i, \tag{4}$$

where f_i, g_i and f_j, g_j are the flux vectors calculated with the flow variables in node i and node j, respectively.

The flux-difference defined by equation (3) can be written as

$$\Delta F_{i,j} = A_{i,j}(U_j - U_i)\Delta s_{i,j} = A_{i,j}\Delta U_{i,j}\Delta s_{i,j}. \tag{5}$$

To define the discrete Jacobian A, the polynomial flux-difference splitting is used. This splitting technique was introduced by the second author [1]. Full details are given in [2, 3]. The polynomial flux-difference splitting is a Roe-type technique , i.e. it satisfies the primary requirements formulated by Roe [4]. However, it is simpler. Its simplicity follows from dropping the secondary requirement of having a unique definition of averaged flow variables. This secondary requirement defines the original Roe-splitting within the class of methods allowed by Roe's primary requirements. However, the secondary requirement is not necessary. The precise splitting used is not really relevant for the method we describe here. Therefore, we do not detail the method any further. The only important result is equation (5).

The discrete Jacobian $A_{i,j}$ in equation (5) has real eigenvalues that are discrete analogs of the normal velocity through the side of the control volume, and the normal velocity plus and minus the velocity of sound. The Jacobian also has a complete set of eigenvectors. These properties are a direct consequence of the hyperbolic character of the Euler equations with respect to time. The Jacobian matrix A can be written as

$$A = R\,\Lambda\,L, \tag{6}$$

where Λ denotes the eigenvalue matrix and where R and L denote the right and left eigenvector matrices in orthonormal form. The matrix A can be split into positive and negative parts by

$$A^{+} = R\Lambda^{+}L\ ,\ \ A^{-} = R\Lambda^{-}L\ . \tag{7}$$

Upwind Flux Definition

For the side ab of the node i and node j control volumes (figure 1), the first order upwind flux is defined by

$$F^{1}_{i,j} \ = \ \frac{1}{2}(F_i + F_j) - \frac{1}{2}(A^{+}_{i,j} - A^{-}_{i,j})\Delta U_{i,j}\Delta s_{i,j}, \tag{8}$$

where

$$F_i = (n_x f_i + n_y g_i)\Delta s_{i,j},\ \ F_j = (n_x f_j + n_y g_j)\Delta s_{i,j}.$$

Using equation (5), equation (8) can be written as

$$F^{1}_{i,j} \ = \ F_i + A^{-}_{i,j}\ \Delta U_{i,j}\ \Delta s_{i,j}. \tag{9}$$

This way of writing the flux shows the incoming wave components.

In order to define a second-order flux, the second part in the right hand side of equation (8), which contains the positive and negative parts of the flux-difference, is decomposed into components along the eigenvectors of the Jacobian, according to

$$\Delta F^{\pm}_{i,j} = A^{\pm}_{i,j}\ \Delta U_{i,j}\ \Delta s_{i,j} = \sum_{n} r^{n}_{i,j}\ \lambda^{n\pm}_{i,j}\ l^{n}_{i,j}\ \Delta U_{i,j}\ \Delta s_{i,j}, \tag{10}$$

where r^n and l^n are right and left eigenvectors of A associated to the eigenvalue λ^n. r^n and l^n are components of R and L, respectively. By denoting the projection of $\Delta U_{i,j}$ on the n^{th} eigenvector by

$$\sigma^n_{i,j} = l^n_{i,j}\ \Delta U_{i,j}, \tag{11}$$

the parts of the flux-difference become

$$\Delta F^{\pm}_{i,j} = \sum_n r^n_{i,j}\ \lambda^{n\pm}_{i,j}\ \sigma^n_{i,j}\ \Delta s_{i,j} = \sum_n \Delta F^{n\pm}_{i,j}. \tag{12}$$

The second-order flux is then defined by

$$F^2_{i,j} \ = \ \frac{1}{2}(F_i + F_j) - \frac{1}{2}\sum_n \Delta F^{n+}_{i,j} + \frac{1}{2}\sum_n \Delta F^{n-}_{i,j} + \frac{1}{2}\sum_n \Delta \tilde{\tilde{F}}^{n+}_{i,j} - \frac{1}{2}\sum_n \Delta \tilde{\tilde{F}}^{n-}_{i,j}, \tag{13}$$

where $\Delta \tilde{\tilde{F}}^{n\pm}$ are limited combinations of the flux-difference components $\Delta F^{n\pm}$ and the shifted differences $\Delta \tilde{F}^{n\pm}$, in the sense of i for positive components and in the sense of j for negative components. The limiter used here is the minmod-limiter. The shifted flux-difference components are constructed based on shifted differences of conservative variables. These are obtained by extending the line segment i, j into the adjacent triangles and constructing intersections with the opposing sides (point i_1 and point j_1 in figure 1). The shifted flux-differences are defined by

$$\Delta \tilde{F}^{n\pm}_{i,j} = r^n_{i,j}\ \lambda^{n\pm}_{i,j}\ \tilde{\sigma}^{n\pm}_{i,j}\ \Delta s_{i,j} = r^n_{i,j}\ \lambda^{n\pm}_{i,j}\ l^n_{i,j}\ \Delta \tilde{U}^{n\pm}_{i,j}\ \Delta s_{i,j}, \tag{14}$$

where $\Delta \tilde{U}^{n\pm}_{i,j}$ denotes the shifted differences. The employed technique to define the second-order flux commonly is called the flux-extrapolation technique. This concept was introduced by Chakravarthy and Osher [5]. For examples illustrating the quality of this second-order formulation on structured grids, the reader is referred to [2, 3, 6].

Boundary Conditions

The examples to follow are either channel flows or flows around airfoils. The internal-type flows have solid, inlet, and outlet boundary conditions. The external-type flows have solid boundaries and far-field boundaries. Figure 2 gives an example of an external flow. The grid generation is detailed in a section below. The far-field boundary is a hexagon with sides 100 chord lengths away from the airfoil.

Inflow and outflow boundary conditions are imposed through the definition of the boundary flux. The boundary flux is calculated by equation (8) with the flux Jacobian $A_{i,j}$ determined with the values of the variables of the boundary node and the difference $\Delta U_{i,j}$ taken as the difference between values in the boundary node and in a ficticious node. The variables in this ficticious node are calculated by the classic extrapolation procedure. At a subsonic inlet, the Mach number is extrapolated while stagnation conditions and flow direction are imposed. For outflow, the stagnation properties and flow direction are extrapolated while the Mach number is imposed.

At solid boundaries, impermeability is imposed by setting the convective part of the flux equal to zero. Thus a special flux definition is used for the boundary edges of the control volumes of boundary

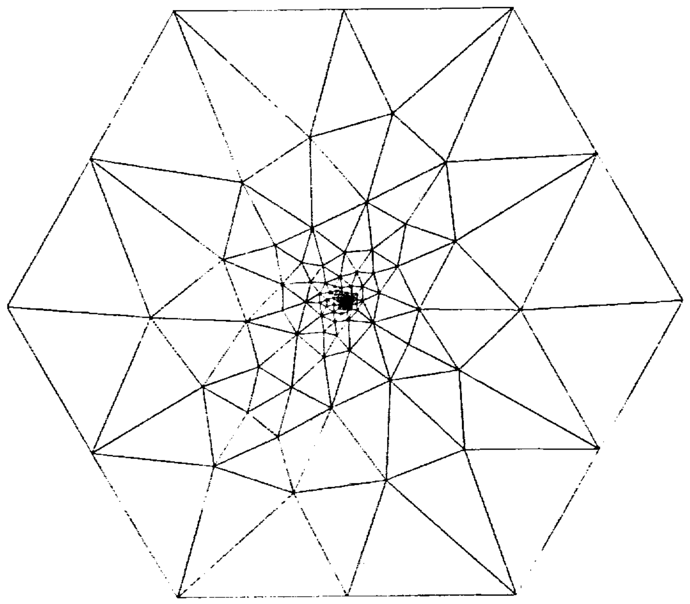

Figure 2: Triangulation around an airfoil with a hexagon far-field boundary 100 chords away from the airfoil. The large distance between boundary and airfoil makes that the airfoil itself is not visible on this figure.

nodes. For a point on the boundary ($i\prime$ in figure 1), the flux through the boundary side can be written as

$$F_{i\prime,j\prime} = F'_{i\prime}, \tag{15}$$

where $F'_{i\prime}$ is the flux calculated with the variables in the node $i\prime$, and where the convective part of the flux is set equal to zero. To give the same appearance to a boundary flux as to an interior flux, all flux expressions can be written as

$$F_{i,j} = F_i + A^-_{i,j}\ \Delta U_{i,j}\ \Delta s_{i,j} + S.O., \tag{16}$$

where $S.O.$ denotes the second-order correction. For a boundary node the point j does not exist. This can be introduced by taking the values of the variables in the ficticious node j equal to the values of the variables in the node i. So, the first order difference of the variables $\Delta U_{i,j}$ and the second-order correction term $S.O.$ in the r.h.s. of equation (16) vanish. The matrix $A^-_{i,j}$ in equation (16) is then calculated with the values of the variables in the node i. The impermeability is introduced in the term F_i. As will be discussed in the next section, the matrix $A^-_{i,j}$ at a solid boundary plays an important role in the relaxation method, although it is multiplied with a zero term.

For an external-type flow, fluxes through edges on the far-field boundary are defined by equation (9). The Jacobian is calculated with the values in the boundary node. The values in the fictitious outside node are taken from the uniform far-field flow. By the flux difference splitting this procedure is equivalent to the use of Riemann invariants.

THE MULTI-STAGE RELAXATION METHOD

In our earlier multigrid formulations for steady Euler equations using structured grids, the Gauss-Seidel relaxation method was always used [2, 3, 6]. Gauss-Seidel relaxation was preferred to Jacobi-relaxation because it has much better smoothing properties (effectiveness of the coarse grid correction) and much better speed of convergence (effectiveness of the relaxation method itself). For structured grid applications the sequential Gauss-Seidel relaxation is very natural. However, it is not simple to use a sequential relaxation method on an unstructured grid, since this requires the construction of paths through the grid. On an unstructured grid, a much more natural method is a simultaneous method instead of a successive one. A simultaneous relaxation method, like the Jacobi relaxation, also has the further advantage of being easily vectorizable and parallelizable. The only drawback is that a simultaneous relaxation method, at least in its basic form, is much less effective than a sequential method.

To repair this, we bring multi-staging into the Jacobi method in the same way as multi-staging is used for time-stepping methods and we use the optimization results with respect to smoothing known for time-stepping schemes. This was first suggested by Morano et al. [7], but not worked out in detail. A more detailed analysis on the possible multigrid performance was made by the authors [8].

We do not intend here to enter a principal discussion on the possibilities of multi-stage Jacobi relaxation. We only want to illustrate that it can be used and that it is sufficiently effective, certainly on unstructured grids, to be attractive in a multigrid context. We choose here a priori the defect correction multigrid procedure as was used in [2, 3, 6]. This means that the second-order part of the flux (S.O. in equation (16)) is updated only on the finest grid and is frozen on all other grids. We choose this configuration mainly for simplicity. The inner, multigrid, iteration cycle is then based on a linear discretization scheme for the partial differential equations. As a consequence of the linearity, this cycle can be rigorously optimized. The outer, defect correction, cycle does not involve any parameters and does not require optimization. In the outer cycle, a second-order accurate flux definition is used. The second-order part of the flux is highly non-linear due to the TVD-limiter. It is certainly possible to use second-order operators in the multigrid formulation as shown in [7] and [8], but it is not clear how to optimize. Therefore, the difficulties associated with non-linear discretization schemes in the multigrid formulation are avoided and only first order discretizations are considered.

For the time-dependent Euler equations, the first order discrete set of equations associated with the node i reads

$$Vol_i \frac{dU_i}{dt} + \sum_j A^-_{i,j}(U_j - U_i)\Delta s_{i,j} = 0, \tag{17}$$

where the index j loops over the faces of the control volume and the surrounding nodes and where Vol_i is the area of the two dimensional control volume. A single-stage time-stepping method with local time-stepping applied to equation (17) gives

$$\left(\frac{Vol_i}{\Delta t_i}\right)(U_i^{n+1} - U_i^n) + \sum_j A^{-n}_{i,j}(U_j^n - U_i^n)\Delta s_{i,j} = 0, \tag{18}$$

where the superscript n denotes the time level.

Using increments $\delta U_i = U_i^{n+1} - U_i^n$, equation (18) is written as

$$\left(\frac{Vol_i}{\Delta t_i}\right)\delta U_i + \sum_j A_{i,j}^{-n}(U_j^n - U_i^n)\Delta s_{i,j} = 0. \tag{19}$$

The Jacobi-relaxation applied to the steady part of equation (17) reads

$$\sum_j A_{i,j}^{-}(U_j^n - U_i^{n+1})\Delta s_{i,j} = 0. \tag{20}$$

Using increments $\delta U_i = U_i^{n+1} - U_i^n$, this gives

$$\left(-\sum_j A_{i,j}^{-}\Delta s_{i,j}\right)\delta U_i + \sum_j A_{i,j}^{-}(U_j^n - U_i^n)\Delta s_{i,j} = 0. \tag{21}$$

The 4x4 matrix coefficient of δU_i in equation (21) is non-singular. In equations (18) to (21), the matrices $A_{i,j}^{-}$ are at the time or relaxation level n. The difference between (single-stage) Jacobi relaxation equation (21) and single-stage time-stepping equation (18) is seen in the matrix coefficient of the vector of increments δU_i. In the time-stepping method, the coefficient is a diagonal matrix. In the Jacobi method, the matrix is composed of parts of the flux-Jacobians associated with the different faces of the control volume. The collected parts correspond to waves incoming to the control volume. In the time-stepping, the incoming waves contribute to the increment of the flow variables all with the same weight factor. In the Jacobi relaxation the corresponding weight factors are proportional to the wave speeds. As a consequence, Jacobi relaxation can be seen as a time-stepping in which all incoming wave components are scaled to have the same effective speed. This is very important for the optimization in the sequel.

For a node on a solid boundary, an expression similar to equation (21) is obtained provided that for a face on the boundary the flux expression of equation (16) is used and that the difference in the first order flux-difference part is introduced as $U_i^n - U_i^{n+1}$,(similar to the term $U_j^n - U_i^{n+1}$ which is used for a flux on an interior face). In order to avoid a singular matrix coefficient of the vector of increments in equation (21), this special treatment at boundaries is necessary. A boundary node can then be updated precisely in the same way as a node in the interior.

To bring in multi-staging is now very simple. For instance, a three-stage Jacobi relaxation is given by

$$\begin{aligned} U_o &= U_i^n \\ U_1 &= U_0 + \alpha_1\,\delta U_i^0 \\ U_2 &= U_0 + \alpha_2\,\delta U_i^1 \\ U_3 &= U_0 + \alpha_3\,\delta U_i^2 \\ U_i^{n+1} &= U_3, \end{aligned}$$

with

$$\left(-\sum_j A_{i,j}^{-l}\Delta s_{i,j}\right)\delta U_i^l = R_i^l = -\sum_j A_{i,j}^{-l}\Delta U_{i,j}^l\Delta s_{i,j}. \tag{22}$$

The parameters α_1,α_2 and α_3 are to be chosen. The increment δU_i^l is obtained from the single-stage Jacobi relaxation method. The method is converted to a multi-stage time-stepping if δU_i^l is replaced by the increment obtained from the single-stage time-stepping method with CFL-number equal to 1. In the sequel, we use the optimization results obtained by Van Leer et al. [9]. For three-stage relaxation, the parameters are $\alpha_1 = 0.1481\ \alpha_3$, $\alpha_2 = 0.40\ \alpha_3$ and $\alpha_3 = 1.5$. Since by the Jacobi relaxation all wave components are scaled to have the same speed, the tuning of the smoothing is correct for all wave components.

To illustrate the performance of multi-stage Jacobi relaxation, we consider a test problem in which no adaption is used. Figure 3 shows the test geometry discretized with two unstructured grids with 10557 and 2657 nodes. The grids have a more or less uniform distribution of the mesh size. Two coarser grids (not shown) with 683 and 182 nodes are also used. The bump has a height of 4% of its chord. The channel height is equal to the bump chord.

Figure 4 shows the iso-Mach line results obtained for a transonic case with an outlet Mach number of 0.79 and for a supersonic case with inlet Mach number of 1.4

Figure 5 shows the convergence history. A full multigrid method with W-cycles is employed. The calculation starts from uniform flow on the coarsest grid. On each level one three-stage Jacobi relaxation is done. One Jacobi stage and one residual evaluation for all nodes on the finest grid, are both counted as one work unit. The corresponding work on a coarse grid is counted proportional to the number of nodes on that grid. A second-order correction is also counted as one work unit. The work associated to intergrid transfer is neglected. Coarse grid points also appear in the finer grids. So injection is used for function value restriction. Defect restriction is obtained through weighting by

$$\frac{R_{i'}}{Vol_{i'}} = \frac{1}{2}\left(\frac{R_i}{Vol_i} + \frac{1}{n}\sum_j^n \frac{R_j}{Vol_j}\right),$$

where the node i on the finer grid corresponds with the node i' on the coarser grid and where j loops over the n neighboring nodes of i. R_i stands for the residual as given by equation (22) and Vol_i is the volume of the control volume. The prolongation is constructed by direct transfer of the coarse grid correction to the corresponding points in the finer grid. Fine grid points that do not correspond to a coarse grid point are given a correction value based on an average of the corrections at the neighboring nodes that do appear on the coarser grid.

The convergence history of the maximum residual is shown in figure 5. The convergence rate is about one order of magnitude per 250 work units. This is an acceptable performance. The best convergence rate on structured grids using Gauss-Seidel relaxation on comparable problems is about one magnitude per 50 work units [2, 3, 6, 10]. Due to the use of a Gauss-Seidel type smoo ther the convergence rate in the structured case is better.

THE MESH GENERATION

The automatic triangulation of an arbitrary set of points can be achieved using Delaunay triangulation. Robust algorithms to construct this triangulation in 2D are available.

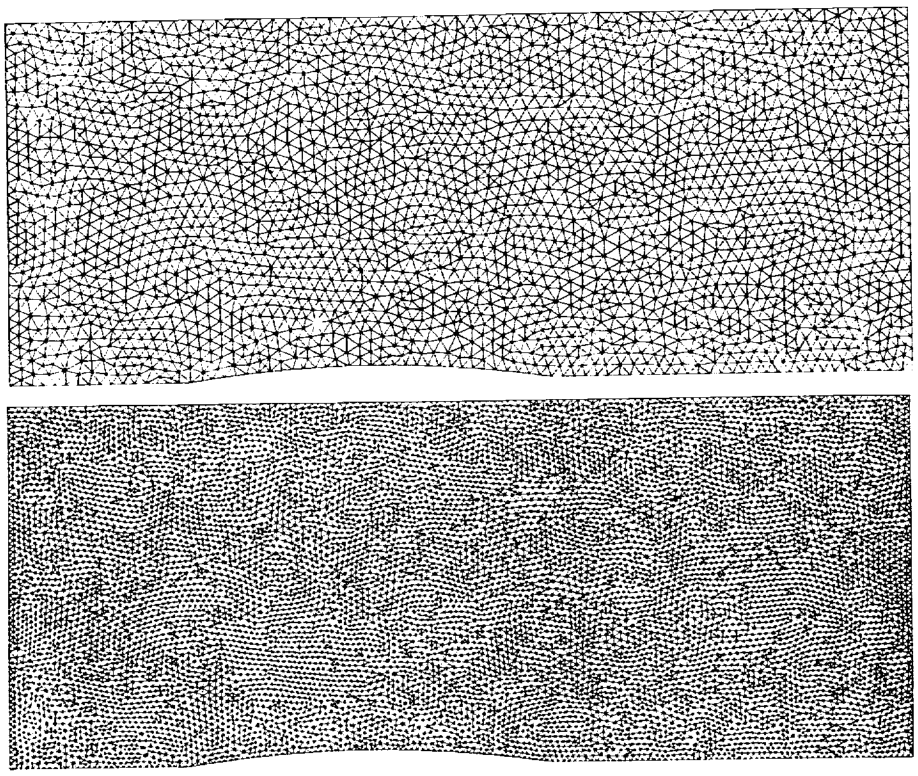

Figure 3: Top : Unstructured grid with 2657 nodes; Bottom : Finest grid with 10557 nodes.

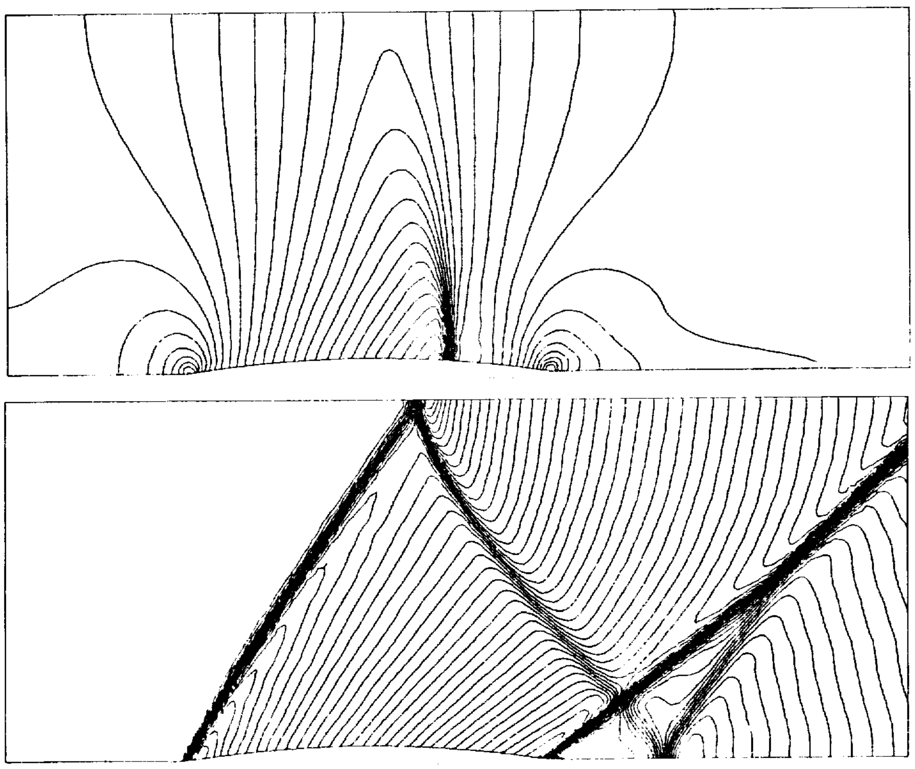

Figure 4: Iso-Mach lines calculated for the transonic test case Mach 0.79 (top) and the supersonic test case Mach 1.4 (bottom). Iso-mach lines per 0.02. Defect correction result with minmod-limiter.

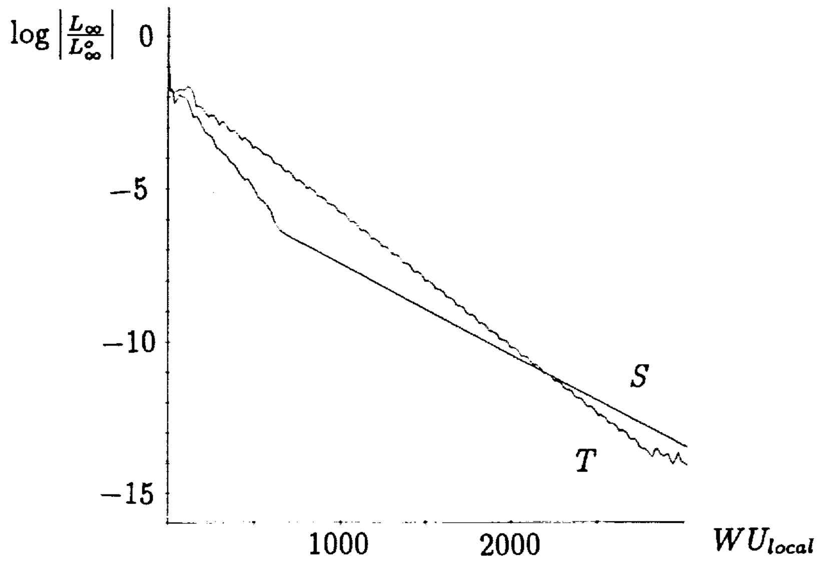

Figure 5: Convergence behaviour for the second-order defect correction scheme (TVD minmod-limiter) using three-stage Jacobi relaxation. Transonic (T) and supersonic (S) test cases.

The grid generator is built with two basic algorithms. The first one is based on the advancing front method, developed by Tanemura et al. [11], simplified to work on a set of nodes which are already connected. The nodes have to lie on the boundary of a polygon. With this algorithm a given, not necessarily convex, polygon can be triangulated. No extra nodes are added. The result is a constrained Delaunay triangulation of the given polygonal boundary. This algorithm was used for the generation of the initial triangulation of the domain. The domain was defined by discretizing the boundaries based on local curvature and local grid spacing.

The second basic algorithm used in the grid generation allows the addition of a node to an existing triangulation. The new node is connected to three or four existing nodes: three nodes if the new node lies inside a triangle, four nodes if the new node lies on an edge. The triangulation is made Delaunay by the use of a diagonal swapping algorithm [12].

For a sequence of multigrid grids, our formulation of the intergrid transfer operators demands that a node of a coarse grid appears in the finer grids. Two strategies can be used to satisfy this condition. A sequence of grids can be constructed by refining a coarse grid (adding nodes), or by coarsening a fine grid (deleting nodes). To add a node to a triangulation the second basic algorithm is used. To delete a node, the node (together with all the connected edges) is removed from the triangulation and then the remaining cavity is retriangulated using the first basic algorithm.

To allow stretching in the construction of the grid, the above described Delaunay triangulation is performed in a transformed space. Every node has it's own transformation parameters, $(\theta, s_{x'}, s_{y'})$. The transformation is a rotation over the angle θ, followed by a rescaling of the new x' and y' axes by $s_{x'}$ and $s_{y'}$. The criteria by which the Delaunay triangulation is built are based on transformed properties.

For the first test case, the bump in a channel of figure 3, multiple grids are built from coarse to fine by adding nodes. An initial grid is generated based on the discretization of the boundary.

Transformation parameters are calculated from boundary mesh spacing and boundary edge direction. The initial grid is refined to form the coarsest grid in the computation. Interior nodes are added in the middle of selected edges, until all edges satisfy an edge length criterion in the locally transformed space. Finer meshes are generated by taking a smaller value of the threshold on the edge length.

The final mesh is smoothed, moving all the nodes of all the grids. It is possible that by this smoothing the triangulation is no longer Delaunay. Therefore, an edge swapping algorithm is used to restore the Delaunay property on all the grids.

For the second test case, the NACA-0012 airfoil, the first multigrid sequence is built in the same way. Then a solution is determined. A new mesh is now generated based on flow adaptive refinement, in which three refinement criteria are used. The first criterion is based on the pressure difference over an edge. If $|p_i - p_j|L_{i,j} \geq L_{ref}\ |p_{max} - p_{min}|/C_p$, then the edge i, j is refined by placing a new node into the center of the edge. In this formulation p_i is the pressure at node i and $L_{i,j}$ is the length of the edge. The variables at the right hand side are minimum or maximum values taken over all the nodes. L_{ref} is a problem dependent reference length and C_p is the sensitivity constant for the pressure criterion. This criterion triggers shockwaves and stagnation regions. The second criterion is based on the entropy difference over an edge. It is similar to the first criterion, however p is replaced by the entropy s. This criterion triggers shock waves and tangential discontinuities. Further, edges are refined where the flow passes through Mach number equal to 1 from supersonic to subsonic flow. This criterion also triggers shock waves.

The adaptively refined mesh can be included directly in the multigrid sequence but large parts of the mesh might coincide with the next coarser mesh. This results in a degradation of the multigrid performance. Therefore, it is better to generate coarser meshes by coarsening the new mesh. The simple criterion, removing a node only if no neighbors of this node are removed, is used. The number of nodes is decreased by roughly 1/4 by doing this. This coarsening is repeated twice to get a coarser mesh.

MULTIGRID RESULTS USING ADAPTIVITY

Starting from the grid shown in figure 2, two successive stages of refinement were done using a threshold value on the edge length in the locally transformed space. Figure 6 shows part of the final grid generated this way. On this grid a solution is calculated using multigrid. By the use of the foregoing adaption criteria, this mesh is refined. Three coarser grids are generated from this grid with the above described coarsening procedure. A solution is calculated using the four grids. The adaption is repeated three times. The solution on the final mesh is given in figure 7 in the form of iso-Mach lines. The NACA-0012 profile has an angle of attack of 1.25 degrees and the Mach number of the incoming flow is 0.80. In total, 4 fine meshes (with for every fine mesh 2 or 3 coarser meshes) were used with 1302 (figure 6) (594,300), 2243 (1256,720,421), 2834 (1577,911,522), 3236 (figure 7) (1782,1010,580) nodes. Figure 8 shows the final grid structure in the shock regions. Figure 9 shows the final grid structure in the leading edge and the trailing edge regions. The refinement of the fourth grid is shown in the left part of figure **??** where only the edges selected for refinement are shown. The right part of figure 10 shows the pressure distribution over the a irfoil.

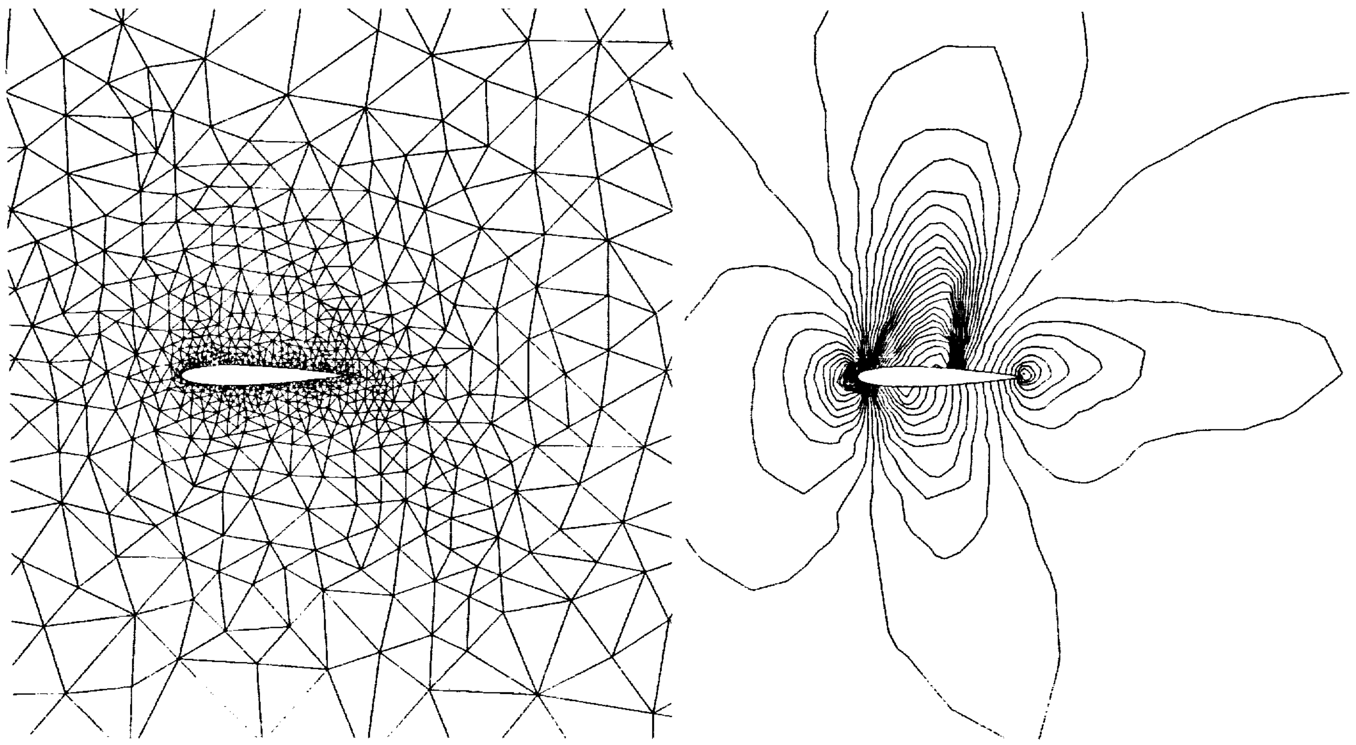
Figure 6: Left: First grid (1302 nodes), Right: Iso-Mach lines per 0.02.

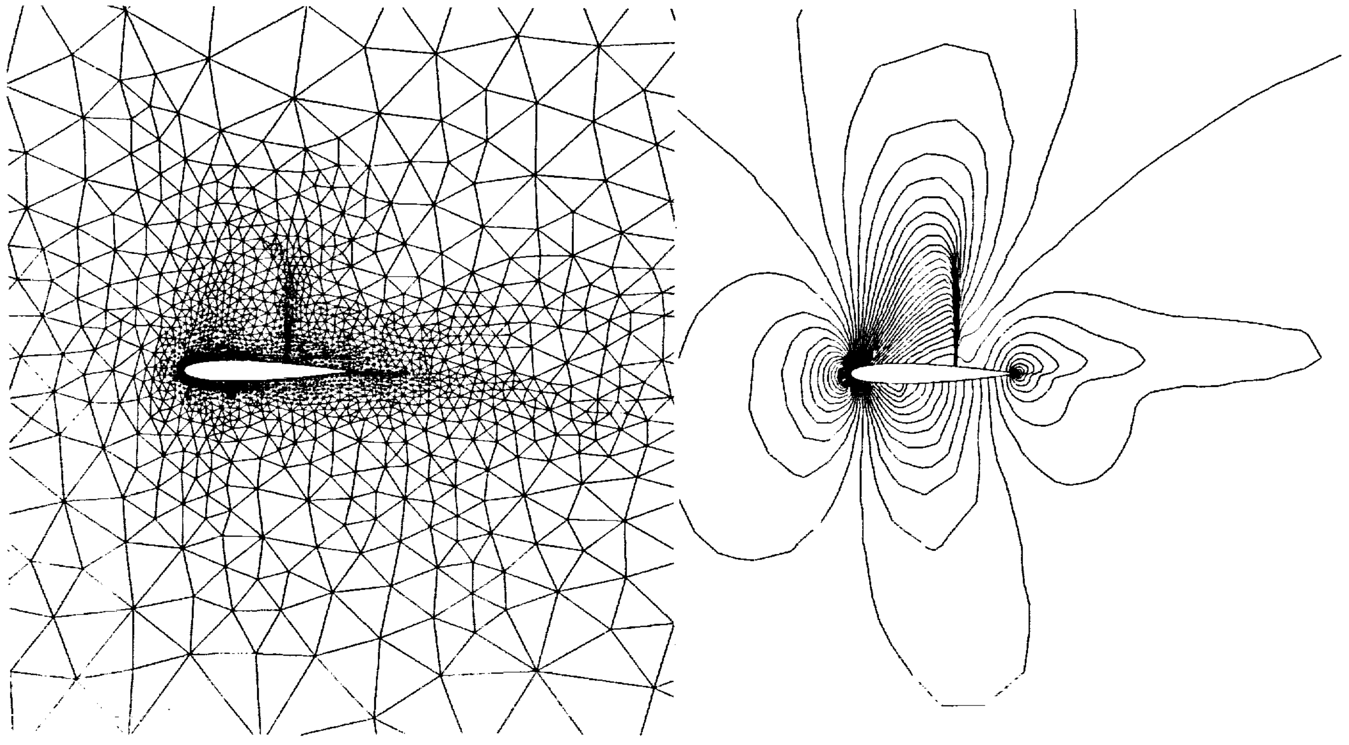
Figure 7: Left: Fourth and final grid (3236 nodes), Right: Iso-Mach lines per 0.02.

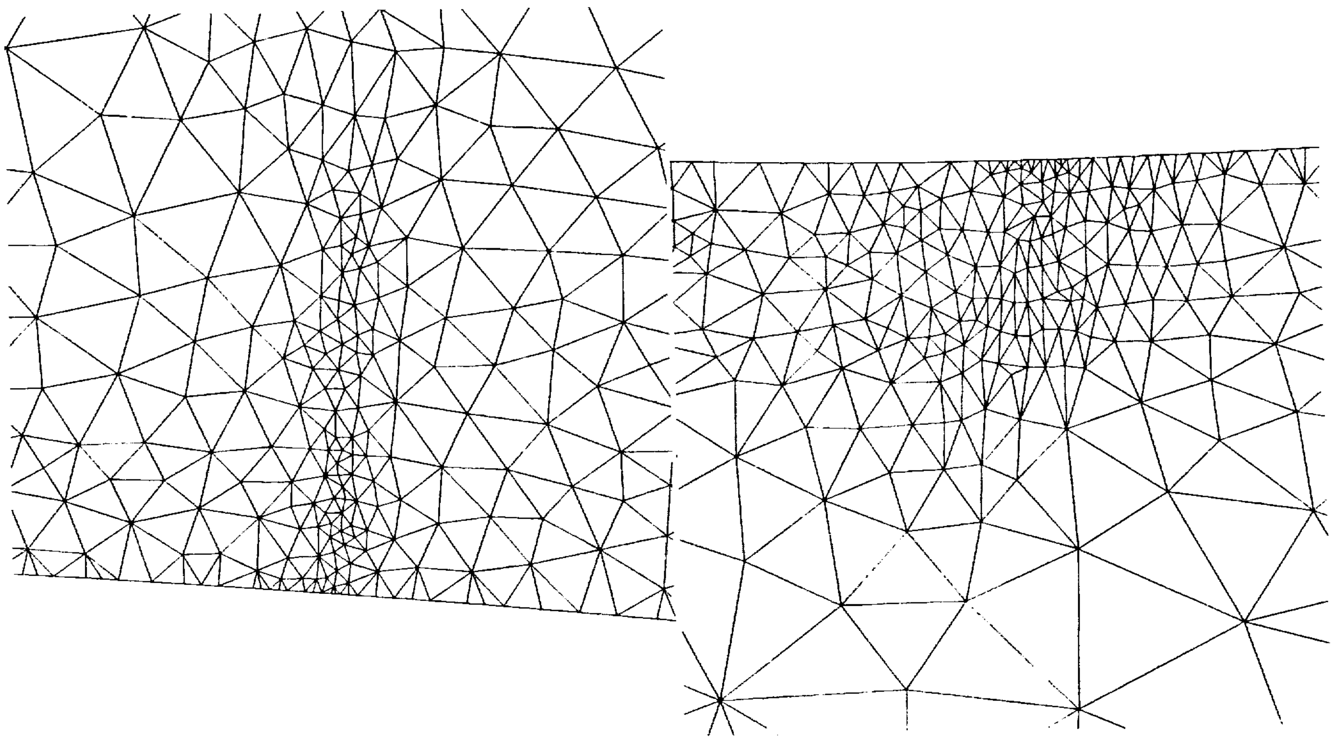
Figure 8: Exploded view of the final grid, shock regions.

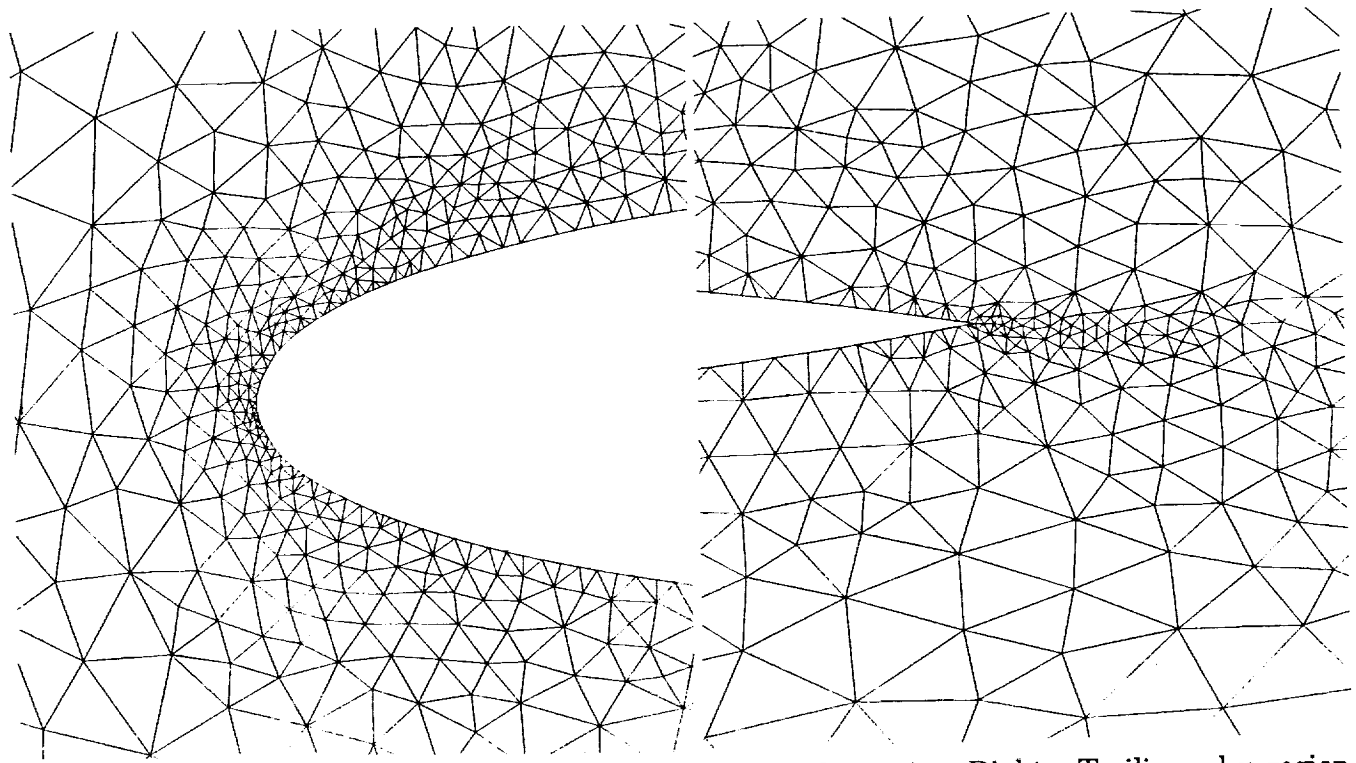
Figure 9: Exploded view of the final grid. Left : Leading edge region; Right : Trailing edge region.

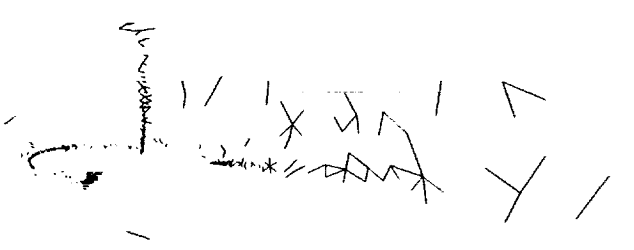
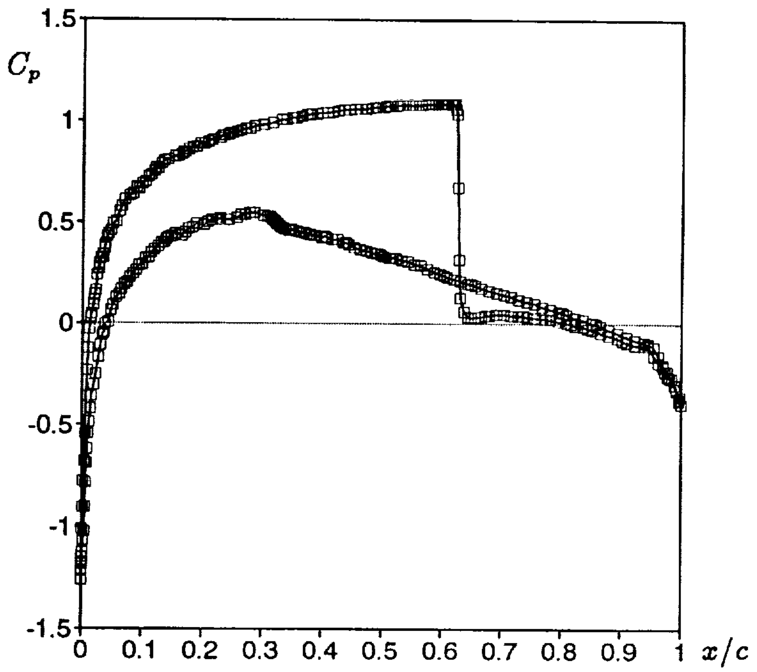

Figure 10: Left : Edges selected for refinement from third to fourth grid; Right : Pressure distribution over the airfoil.

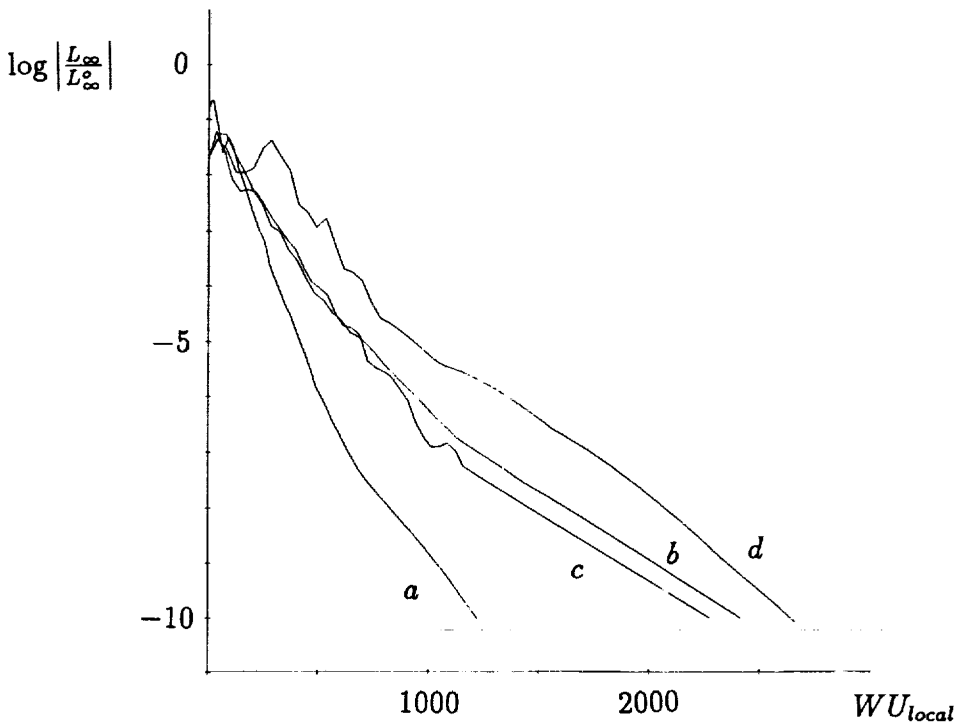

Figure 11: Convergence behaviour for the second-order defect correction scheme (TVD minmod limiter) on the NACA-0012 test case, using three-stage Jacobi. Logarithm of the residual as function of local work units. Multigrid with fine grids 1302 (a), 2243 (b), 2834(c), and 3236 (d) nodes.

Figure 11 shows the convergence behaviour. For each multigrid phase, the work unit is defined based on the number of nodes in the finest grid during this phase. This way of representing the convergence is chosen to demonstrate the mesh independency of the convergence.

ACKNOWLEDGEMENT

The research reported here was granted under contract IT/SC/13, as part of the Belgian National Programme for Large Scale Scientific Computing and under contract IUAP/17 as part of the Belgian National Programme on Interuniversity Poles of Attraction, both initiated by the Belgian State, Prime Minister's Office, Science Policy Programming.

REFERENCES

[1] Dick, E. (1988). A flux-difference splitting method for steady Euler equations. *J. Comp. Phys.*, *76*, 19-32.

[2] Dick, E. (1990). Multigrid formulation of polynomial flux-difference splitting for steady Euler equations. *J. Comp. Phys.*, *91*, 161-173.

[3] Dick, E. (1991). Multigrid methods for steady Euler and Navier-Stokes equations based on polynomial flux-difference splitting. *Multigrid Methods III*, Int. Series on Numerical Mathematics, *98*, Birkhauser Verlag, Basel, 1-20.

[4] Roe, P. L. (1981). Approximate Riemann solvers, parameter vectors and difference schemes. *J. Comp. Phys.*, *43*, 357-372.

[5] Chakravarthy, R. S., Osher, S. (1985). A new class of high accurate TVD schemes for hyperbolic conservation laws. *AIAA paper*, *85-0363*.

[6] Dick, E. (1991). Multigrid solution of steady Euler equations based on polynomial flux-difference splitting. *Int. J. Num. Methods Heat Fluid Flow*,*1*,51-62.

[7] Morano, E., Lallemand, M.-H., Leclerq, M.-P., Steve, H., Stoufflet, B. and Dervieux, A. (1991). Local iterative upwind methods for steady compressible flows. *GMD-Studien*, *189*.

[8] Dick, E., Riemslagh, K. (1993). Multi-stage Jacobi relaxation as smoother in a multigrid method for steady Euler equations. *Proc. of ICFD Conference on Numerical Methods for Fluid Dynamics*, Reading, to appear.

[9] Van Leer, B., Tai, C. H. and Powell, K. G. (1989). Design of optimally smoothing multi-stage schemes for the Euler equations. *AIAA-paper*,*89-1933*.

[10] Hemker, P. W. and Spekrijse, S. P. (1986). Multiple grid and Osher's scheme for the efficient solution of the steady Euler equations. *Appl. Numer. Math.*, *2*, 475-493.

[11] Tanemura, M., Ogawa, T., Ogita, N (1983). A new algorithm for three-dimensional Voronoi tessellation. *Journal of Computational Physics*, *51*, 191-207.

[12] Lawson, C. L. (1972). Generation of a triangular grid with application to contour plotting. *California Institute of Technology, Jet Propulsion Laboratory*, Technical Memorandum, 299.